MODELING OF DEEP IN-MEMORY ARCHITECTURE (DIMA) IC FOR DNNS

By

Alkin Ozyapici

Senior Thesis in Electrical Engineering

University of Illinois at Urbana-Champaign

Advisor: Naresh R Shanbhag

May 2021

Abstract

The Deep In-Memory Architecture IC from Prof Shnabahg's group is studied in this paper. DIMA embeds dot product computation in the periphery of the memory and as a result leady to energy and latency cost savings compared to conventional architectures. However, DIMA suffers from accuracy drops due to manufacturing and architecture related non-idealities. A functional model of the DIMA IC was Implemented in Python utilizing analytical models to simulate the effects of threshold voltage variations of a 6T SRAM cell and other non-idealities in order to simulate the accuracy of Deep Neural Networks (DNNs) on DIMA. Two DNNs are considered: VGG11 and Resnet20. The DNNs were tested on the CIFAR10 and CIFAR100 datasets to see the effects of different classification tasks. While the floating-point accuracy for VGG11 and Resnet20 with CIFAR10 dataset was 92.12% and 91.20% respectively, when the networks were mapped on DIMA the accuracy dropped to 87.03% and 77.42% respectively. It can be seen from the results that DNNs suffer accuracy loss when mapped on DIMA due to quantization and additional non-idealities of the architecture. To compensate for the circuit specific noises the studied DIMA IC also includes an on-chip trainer. On-chip training was simulated by retraining the networks after they were mapped on DIMA. After on-chip training, it was seen that DIMA can achieve comparable if not better accuracies than the baseline 6-bit fixed-point accuracy with both CIFAR-10 and CIFAR-100 datasets for both DNNs. It was also observed that the accuracy drop-off was more significant when the networks were simulated on DIMA with CIFAR100 dataset, from 68.74% to 46.37% for VGG11 before on-chip training. It can be seen that the noise related to DIMA effects the complex 100-class task of CIFAR100 dataset more significantly than the 10-class classification in CIFAR10.

Subject Keywords: in-memory processing; machine learning; accelerator; modeling; PyTorch

Acknowledgments

First of all, I would like to thank my advisor Professor Naresh Shanbhag for providing this opportunity to work under him and providing his guidance throughout the whole year. I also would like to thank Hassan Dbouk, my graduate advisor, for providing guidance in our weekly meetings and helping me throughout the whole process.

Contents

1. Introduction

Recently data analytics and machine-learning applications have been taking a bigger space in computing with increasing interest to implement these in edge devices such as wearables and IoT devices. However, these edge devices are heavily resource-constrained due to their compact form factor and finite source of energy. Therefore, energy efficiency is one of the top priorities while designing machine learning processors for edge applications. Machine learning algorithms require processing large volumes of data (activations and weights) and therefore their energy requirements are highly correlated with energy costs of a data access. Moreover, the inference latency of these algorithms is dominated by the data access times as well. The Deep In-Memory Architecture (DIMA) analyzed in this paper was developed by Prof. Naresh Shanbhag's research group and addresses the need for an energy-efficient machine-learning platform for edge devices by embedding the processing engine in the memory. This alone leads to significant energy gains as well as delay reduction as it significantly reduces the data access energy and delay costs since the computation is done in the memory itself. As the computation is done in the memory, there are no significant delay or energy costs relating to memory busses between the processor and the memory. It was shown [2] that this implementation leads to 10x and 5.3x energy and delay reduction respectively, resulting in a total 53x energy-delay product reduction compared to a baseline digital architecture. This thesis aims to model the DIMA and its non-idealities using PyTorch and measure the accuracy of different DNNs while simulated on the modeled DIMA. Chapter 2 provides information about the DIMA architecture and comparisons between DIMA and conventional designs. Chapter 3 goes over the source of non-idealities and the analytical models used to characterize them. Chapter 4 explains the DNNs and datasets used for testing and the methodology. The acquired results are presented in Chapter 5 and Chapter 6 provides a conclusion.

2. Literature Review

2.1 Traditional Approach to Fixed-Point Architectures

Conventional digital processors (Figure 2.1 (b)) today use a von Neuman architecture where the memory and the processor are separate, and they are connected through large memory busses and controllers. With this architecture, reading data from memory and transferring it to the processor results in large energy and delay costs since the memory busses are large capacitive loads.

Further, in a conventional SRAM array, the data is stored in a row-major format and one bit per column is read per bit line pre-charge. The realization of a 0 or 1 bit is done by the sense amp on the bit lines. Therefore, in this classical approach reading a single word, B_D, of length D, the SRAM needs to perform D bit line pre-charges. Moreover, in a conventional SRAM only N/L words can be read due to column muxing, where N is the width of the SRAM array and L is the column muxing ratio. Overall to read N words of length D, the SRAM cell needs to be pre-charged LD times. The capacitance associated with the bit lines are high as the line is connected to the whole column of cells, and every pre-charge cost significant energy. However, for these architectures, there is no additional noise besides the quantization noise for fixed point computation.

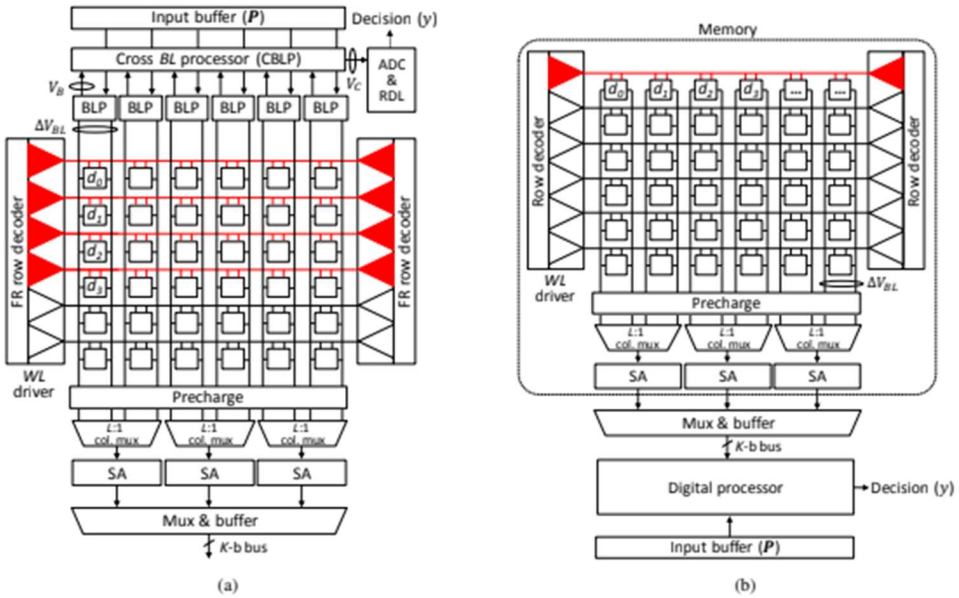

Figure 2.1: A comparison of a) Deep in Memory Architecture with Functional Read (FR) and b) conventional digital architecture [2]

2.2 DIMA Architecture

DIMA architecture (Figure 2.1 (a)) [2] was designed to eliminate the large energy and delay costs associated with the conventional memory read. To do so, Functional Read is introduced (FR) alongside mixed-signal processors embedded in the periphery of the memory array. In the specific prototype that is studied in this thesis, the DIMA IC uses a standard 6T SRAM cell (Figure 2.2).

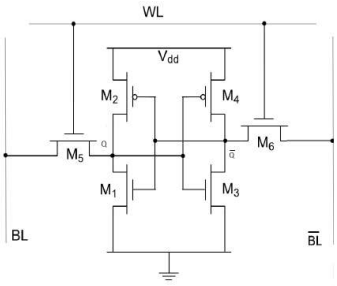

Figure 2.2: Transistor Schematic of a Standard 6T SRAM Cell

For reading a word from the memory DIMA utilizes a method called FR to read a word, B_D, of length D unlike the conventional approach. For the FR stage, DIMA stores the values in a column major format which enables the entire word to lie on the same bit line. To perform the FR word line voltage V_{WL} is modulated with respect to the significance of the bits. This results in a ΔV_{BL} proportional to the decimal value of the word B_D. Utilizing the functional read DIMA can read a word B_D in one bit line pre-charge and saving energy over a conventional SRAM array. Moreover, the FR method enables DIMA to read as many words as the width of the SRAM array at the same time with the elimination of column muxing, leading to massive parallelism over conventional SRAM. The ΔV_{BL} values are multiplied with the words passed from the input buffer in the bit-line-processor (BLP) parallelly. Then the cross-bit-line-processor (CBLP) can perform different analog computations on the BLP outputs, such as addition to perform a dot product. Lastly, the analog result is fed into an ADC to acquire the digital result. The simplified data flow is shown in Figure 2.3.

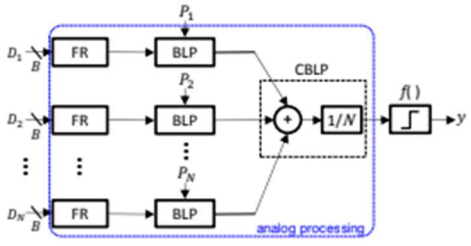

Figure 2.3: DIMA Data Flow [2]

While the specific implementation of the processor in DIMA and the FR method leads to high levels of parallelism and energy savings, it is also susceptible to noise in the SRAM array, specifically to variations in threshold voltage, V_t. However, it was shown that the architecture is robust, and the DIMA architecture can perform support vector machine operations with comparable accuracy to conservative architectures [2]. Moreover, the studied IC has an on-chip trainer to adapt the values to the instance-specific circuit parameters such as the threshold voltage to mitigate the accuracy drops rising from non-idealities inherent to the architecture. Based on these findings, it is expected that DIMA can perform well with neural network applications. The goal of this thesis is to verify this hypothesis.

3. DIMA Non-Idealities

3.1 ΔV_{BL} Variation due to Threshold Voltage Variations

In the FR stage every cell is discharged individually, and the access transistors are turned on according to their bit significance. The discharge path for V_{BL} can be modeled as in Figure 3.1, where C_{BL} is the bit-line capacitance and V_{WL} is the word-line voltage. From the circuit model and using the alpha cell current equation for a transistor we achieve (3.1) for the discharge current, where k'_n is the process transconductance and α is the fitting parameter. The voltage change for one cell on the bit-line ΔV_{BL} using this model is as in (3.2) for pulse width T for V_{WL}. For a word B_D, of length D, the total ΔV_{BL} is acquired in (3.3) where T_{min} is the pulse width for V_{WL} of the least significant bit in the word.

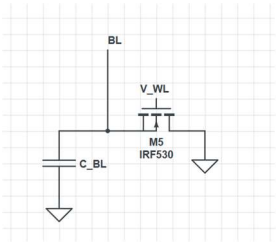

Figure 3.1: The discharge path for a single 6T SRAM cell

$$I_{cell} \approx \frac{W}{L} k'_n (V_{WL} - V_t)^{\alpha} \quad (3.1)$$

$$\Delta V_{BL} = \frac{I_{cell} T}{C_{BL}} \quad (3.2)$$

$$\Delta V_{BL}(total) = \frac{I_{cell} T_{min}}{C_{BL}} \sum_{m=0}^{D-1} 2^m \overline{d_m} \quad (3.3)$$

However, during manufacturing process the threshold voltage of transistors are prone to variations and V_T is not a constant. The non-ideal threshold voltage can be modeled with an additive Gaussian noise model as is in (3.4). The threshold voltage for a specific transistor is determined at manufacturing and is static once the chip is manufactures. Using the non-ideal threshold voltage in (3.4) one can derive the non-ideal cell current for the discharge path (3.5). Using first order Taylor expansion the variation in the non-ideal cell current can be modeled with the additive noise model in (3.6). When we combine the non-ideal cell current equation and the ΔV_{BL} equation we reach the non-ideal voltage difference in the bit-line ΔV_{BL} (3.7).

$$V_T = V_t - v_t, where\ v_t \sim \mathcal{N}\left(0, {\sigma^2}_{v_t}\right), \sigma_{v_t} \approx 23.8mV \quad (3.4)$$

$$I + i = \frac{W}{L} {k'}_n \left(V_{WL} - (V_t - v_t)\right)^{\alpha} \quad (3.5)$$

$$\sigma_i = I\left(\frac{\alpha \sigma_{v_t}}{V_{WL} - V_t}\right) \quad (3.6)$$

$$\Delta V_{BL}(total) = \frac{T_{min}}{C_{BL}} \sum_{m=0}^{D-1} 2^m \overline{d_m}(I_{cell} + i) \quad (3.7)$$

3.2 Clipping Noise due to Finite V_{BL} Pre-Charge Voltage

The equation for ΔV_{BL} (3.7) is acquired above. However, these equations assume that the V_{BL} pre-charge voltage is high enough to realize all ΔV_{BL} values. This is not the case in reality, and some high ΔV_{BL} values cannot be realized since V_{BL} pre-charge voltage is finite. This will result in the clipping of the high values that are stored in the SRAM. The clipping can be shown in (3.8).

$$\Delta V_{BL} = \frac{T_{min}}{C_{BL}} \sum_{m=0}^{D-1} 2^m \overline{d_m}(I_{cell} + i)\ if\ \Delta V_{BL} < V_{BL}(precharge), else\ \Delta V_{BL} = V_{BL}(precharge) \quad (3.8)$$

3.3 Accumulation of Quantization Noise due to Finite Width of SRAM Array

After the FR step is completed, the data is processed in the BLP and the CBLP. However, there are only N number of BLPs, where N is the width of the SRAM array, and one FR cycle can only process N words and the output is quantized in the ADC. In a specific case of a 512-length dot-product, for example the first two fully connected layers in VGG-11, the result will be calculated in two FR stages and then the two results will be accumulated. This leads to the accumulation of quantization noise in the results, and the noise grows as the dot product grows longer.

4. Methodology

4.1 The Deep Neural Network Models

Two networks have been chosen to test and benchmark the models in this thesis. The first one is VGG-11 [6], a conventional multi-layer convolutional DNN. The second one is Resnet20 [5] which is a residual network. The network topologies and the visual representation is provided in Figure 4.1 and Figure 4.2.

Figure 4.1: Network Topology of VGG-11

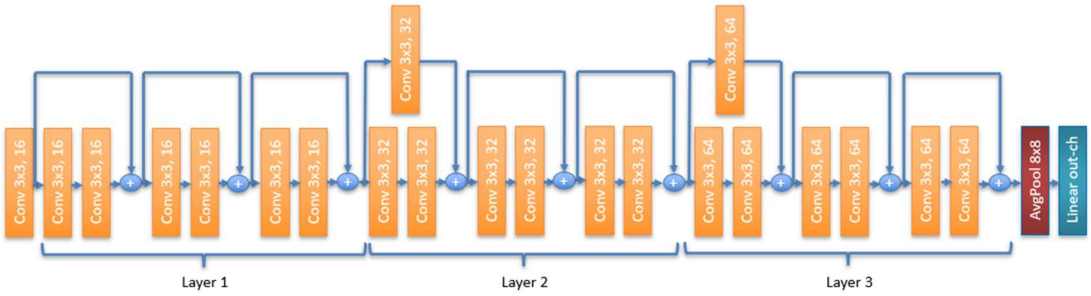

Figure 4.2: Network Topology of Resnet20

These two networks have very different architectures as well as the complexity. VGG-11 is significantly larger in terms of number of parameters, and it leads to a larger number of MAC operations for a 32x32 input image, which is used in testing. The complexities of these networks are given in Table 4.1.

Table 4.1: Network Complexities

	# of Parameters	# of MACs
VGG-11	9.799M	153.643M
Resnet20	278K	41.220M

4.2 Datasets

CIFAR-10 and CIFAR-100 datasets [4] were used to train and test the networks defined above. The CIFAR datasets were chosen since they are widely used in the field, and there are several benchmarks for the datasets that we can compare our DIMA accuracies.

CIFAR-10 is a 10-class dataset with 60000 32x32 images, 50000 training images and 10000 test images. The training images are divided into five batches of 10000 images, and the test set is one single batch with 10000 images. Both the training set and the test set contains equal number of images from each class. The classes are mutually exclusive.

The CIFAR-100 dataset contains the same images as CIFAR-10, however, they are divided into 100 classes with 6000 images per class. Again, there are 50000 training images and 10000 test images, and the classes are mutually exclusive. The images are first divided into 20 superclasses and then into 100 specific classes, and they come with a “coarse label”, the superclass they belong to, and a “fine label”, the specific class they belong to. However, in this thesis, only the fine labels were used, and the task was a 100-class classification task.

4.3 Training-Testing Methodology

VGG-11 and Resnet20 were trained from scratch to get the benchmark floating-point accuracies. The training parameters in Table 4.2 were used to train the networks. After the networks are trained, the weights, inputs and outputs were quantized with 6-bit precision and the accuracy for a 6-bit conservative architecture was acquired.

Table 4.2: Floating Point Training Parameters for the Networks

Training Parameters	Value
Learning Rate	0.01
Weight Decay	5e-4
Momentum	0.9
# of Epochs	200
Scheduler	Cosine Annealing

The floating-point weights were also used in the networks that were simulated with the DIMA parameters. All the non-idealities that were described in Chapter 3 were modeled in software and simulated. The results were acquired averaging 15 independent runs since the non-idealities are random variables and can show significant differences between two instances. After the initial results were acquired the DIMA networks were trained again to simulate the effects of the on-chip trainer. The on-chip training parameters are listed in Table 4.3. The overall testing methodology is described in Figure 4.3.

Table 4.3: The On-Chip Training Parameters

Training Parameters	Value
Learning Rate	0.001
Weight Decay	None
Momentum	None
# of Epochs	20
Scheduler	lr = lr/2 every 7 epoch

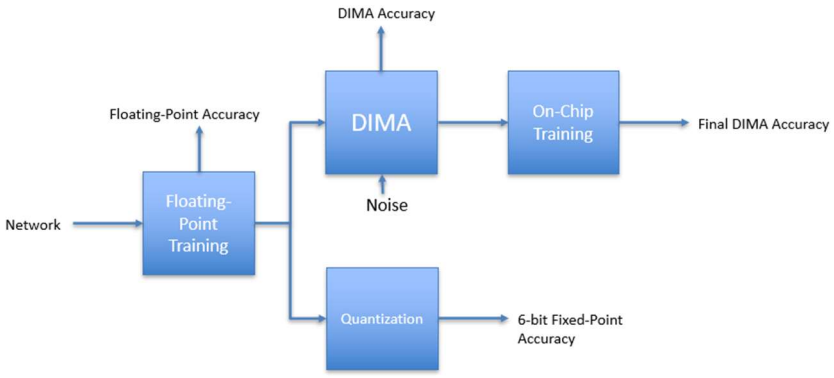

Figure 4.3: The Training and Testing Methodology

5. Results

- Floating Point Accuracy:

Floating-point accuracy of Resnet20 and VGG-11 are 91.30% and 92.12% respectively for the CIFAR-10 dataset and are very similar with less than 1% difference. This is not the case for the CIFAR-100 dataset as, VGG-11 performs nearly 3% better than Resnet20. The floating-point accuracies are presented in Table5.1.

Table 5.1: Floating Point Accuracy Results

Dataset/Network	VGG-11	Resnet20
CIFAR-10	92.12%	91.30%
CIFAR-100	68.74%	65.82%

- 6-bit Quantized Accuracy:

After 6-bit quantization the difference in accuracy between VGG-11 and Resnet20 is more significant. While VGG-11 can achieve an accuracy within 1% of the floating-point accuracy, Resnet20 suffers more significant accuracy drop to 84.49% with CIFAR-10 dataset. The accuracy drops are more severe for both DNNs when CIFAR-100 dataset is used. VGG-11's accuracy drops nearly by 10% to 59.28%, and Resnet20's accuracy drops 25.57% suffering a 40% drop. The 6-bit quantized accuracies are presented in Table 5.2.

Table 5.2: 6-bit Quantized Accuracy Results

Dataset/Network	VGG-11	Resnet20
CIFAR-10	90.97%	84.49%
CIFAR-100	59.28%	25.57%

- DIMA Accuracy:

When the networks are simulated with the DIMA layers the results are similar to the 6-bit quantized accuracies, but the accuracy drop-off is more significant because of the DIMA specific non-idealities. VGG-11's accuracy falls to an average of 87.03% for CIFAR-10, 3% lower than the 6-bit quantized accuracy. Resnet20's accuracy falls to an average of 77.42% for CIFAR-10, 7% lower than the 6-bit quantized accuracy. For CIFAR-100 the results show the same trend as CIFAR-10 with VGG-11 and Resnet20 achieving around 46% and 20% respectively. The results are presented in Table 5.3.

Table 5.3: DIMA Accuracy Results

Dataset/Network	VGG-11	Resnet20
CIFAR-10	87.03±0.47%	77.42±2.14%
CIFAR-100	46.37 ± 1.96%	20.84±3.69%

- DIMA Accuracy After On-Chip Training:

VGG-11 achieves an accuracy very similar to the 6-bit quantized accuracy after on-chip training with an average of 90.72% for CIFAR-10 dataset. For the CIFAR-100 dataset, VGG-11 accuracy exceeds the 6-bit quantized accuracy with an average of 65.02%. Resnet20 accuracies for both CIFAR-10 and CIFAR-100 exceed the 6-bit quantized accuracy with an average of 88.62% and 54.79% for respective datasets. The results after on-chip training are presented in Table 5.4.

Table 5.4: DIMA Accuracy Results After On-Chip Training

Dataset/Network	VGG-11	Resnet20
CIFAR-10	90.72±0.11%	88.62±0.32%
CIFAR-100	65.02±0.34%	54.79±1.64%

6. Conclusion

From the results in Chapter 5, it can be seen that VGG-11 and Resnet20 behave very differently when noise is introduced. It is evident that VGG-11, as the network with significantly higher number of parameters, is less susceptible to noise and shows superior performance to Resnet20 in every given task even though the initial floating-point accuracies are similar.

Overall, it is seen that networks suffer significant loss when they are mapped onto DIMA compared to the floating-point results. However, after on-chip training is simulated, the networks can achieve similar if not better results than a 6-bit fixed point architecture. This proves that DIMA can be used as a viable alternative to digital fixed-point architectures in resource constrained environments, while providing significant energy gains.

In the future, we aim to enhance the software models that were used in this research to include energy models for the DIMA as well to find the consumption for specific tasks. The energy models will enable us to analyze different networks from an energy perspective and see the tradeoffs of accuracy and energy gains when they are mapped on to DIMA.

References

[1] S. K. Gonugondla, M. Kang and N. R. Shnabhag, “A Variation-Tolerant In-Memory Machine Learning Classifier via On-Chip Training” *IEEE Journal of Solid-State Circuits*, vol. 53, no. 11, 2018.

[2] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, “A Multi-Functional In-Memory Inference Processor Using a Standard 6T SRAM Array,” *IEEE Journal of Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, 2018.

[3] M. Kang, S. K. Gonugondla, M.S. Keel, and N. R. Shanbhag "An Energy-Efficient Memory-Based High-Throughput VLSI Architecture for Convolutional Networks" *IEEE Int. Conf. Acoust. Speech Signal Process.* pp. 1037-1041 May 2015.

[4] A. Krizhevsky, “Learning Multiple Layers of Features from Tiny Images”, Master’s Thesis, Department of Computer Science, University of Toronto, 2009.

[5] K. He, X. Zhang, S. Ren, J. Sun “Deep residual Learning for Image Recognition”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[6] K. Simonyan, A. Zisserman “Very Deep Convolutional Networks for Large-Scale Image Recognition”, *International Conference on Learning Representations*, 2015.